How to Configure NIVS WirelessHART Development Kit for Modbus TCP

Rushad Antia

In order for your sensors to have their values mapped to the Modbus registers you must have configured them as instructed in ‘How to Configure VS220 Sensor Modules to Report Sensor Values’.

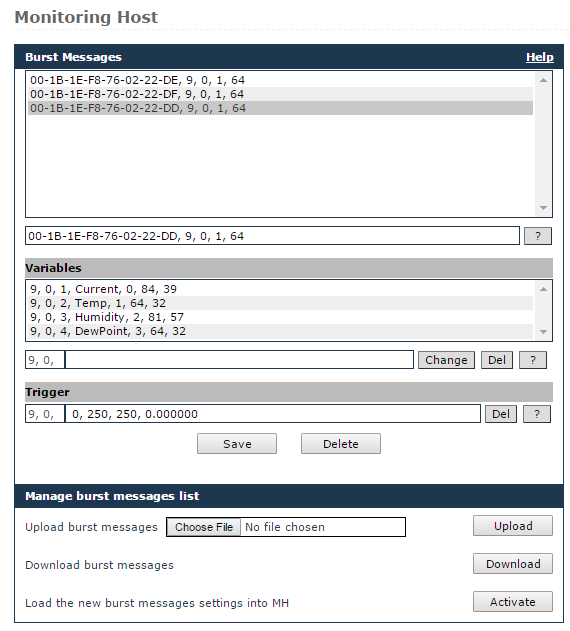
Necessary Tools:

ModbusRegisterGenerator.jar (<https://github.com/rush2sk8/tesim_nivis_setup/tree/Rushad>)

TF6250-Modbus-TCP.exe - (<http://www.beckhoff.com/forms/twincat3/warenkorb.aspx?lg=en&title=TF6250-Modbus-TCP&version=1.0.53.0>)

# Mapping Sensor Values to Modbus Input Registers

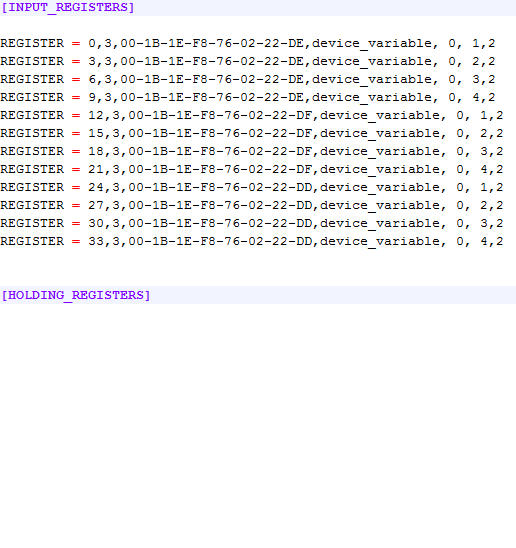
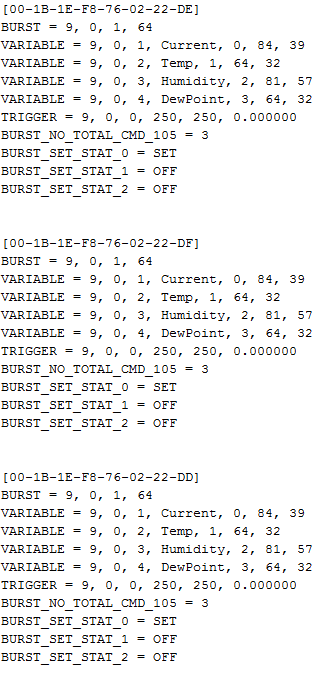
Firstly go to your Monitoring Control System found at http://<VR\_IP>/app/mhconfig.html and download the burst messages.



Save this file to the location where the ModbusRegisterGenerator.jar file is located.

Next launch the ModbusRegisterGenerator.jar and click “Yes”. This program should export a file called “modbus\_gw.ini”.

Below is an example of the generated file:

 Monitor\_Host\_Publishers.conf 🡺 modbus\_gw.ini

2

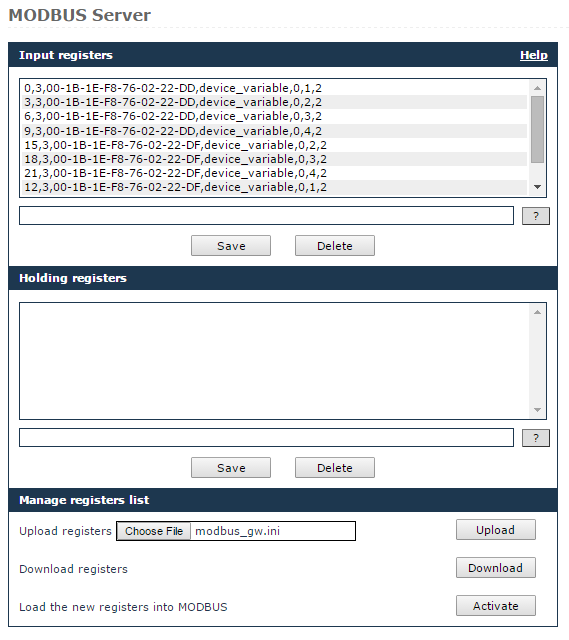
1

1. The registers are formatted as such :  *{<START ADDRESS>, <WORD COUNT>, <EUI64>,<REGISTER TYPE>, <BURST MESSAGE>, <DEVICE VARIABLE CODE>,<DEVICE STATE>}*

* Start Address: Represents the start address (previous address + word count) of the Input Register in the Modbus Server.
* Word Count: Represents the number of data words to read. For the default sensors it is 3 because that is how much it needs for a floating point value.
* EUI64: The MAC address of the sensor.
* Register Type: Should be left as is
* Burst Message: The index of the burst message which can be found as the second element in the Monitor\_Host\_Publishers.conf file under VARIABLE.
* Device Variable Code: The burst message variable to map this register to.
* Device State (uses 2 by default) :
  + 0 – The Wireless Hart device variable is mapped onto the chunk of registers, starting from address <start\_addr>. The value should be interpreted as 4 bytes float. Wireless Hart Communication errors (e.g. device not joined) will make addressing these registers return the MODBUS exception 0x04 (slave device failure);
  + 2 – The Wireless Hart device variable is mapped onto the chunk of registers, starting from address <start\_addr>+1. The value should be interpreted as 4 bytes float. At address <start\_addr> there is a “device state register” generated by the MODBUS server which reflects the state of the Wireless Hart communication. The value should be interpreted as 16 bit big-endian integer with the following values:
    - 128: Device Joined with fresh data
    - 8: Device NOT joined and no data read so far
    - 20: Device NOT joined but with stale data
    - 24: Device Joined but no data read so far
    - 4: Device Joined but with stale data

1. Notice how the numbers [1, 4] on both files are identical. This means that those variables are mapped to the first number in the line. For example in the files above the current variable of 00-1B-1E-F8-76-02-22-DE is mapped to register 0. Every subsequent variable will be mapped to a register that is the previous register plus the word count, which in this case is 3.

Now that you know how the Register file works we can now upload it to the gateway. Go to your MCS and head to the MODBUS tab or http://<VR\_IP>/app/modbus.html . Click on “Choose File” and select the generated “modbus\_gw.ini” file, and click “Upload” then “Activate”.



1

2

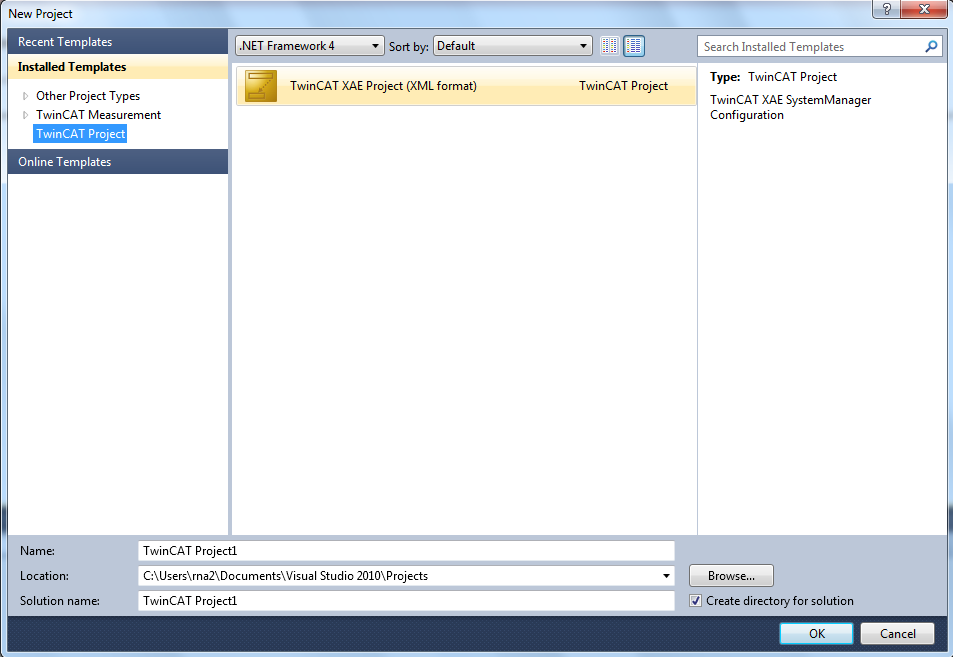
3

☺CONGRATULATIONS ON SETTING UP YOUR MODBUS REGISTERS☺

# Polling Sensor Data through Modbus TCP on a Beckhoff PLC

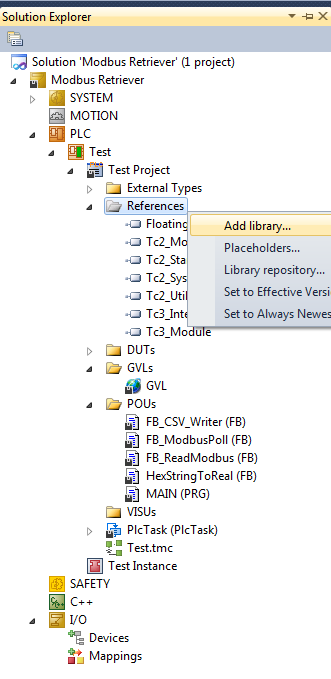
Firstly make sure you have installed TF6250-Modbus-TCP.exe on your PC and the PLC.

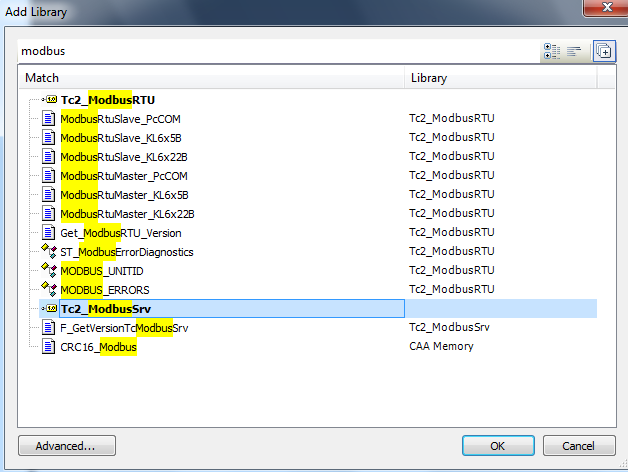
Create a new TwinCAT XAE Project



Next download the PLC Modbus code from <https://github.com/rush2sk8/tesim_nivis_setup/tree/Rushad/Setting%20up%20modbus/PLC%20Code>

In order for these blocks to work you must first add the Modbus library by going to Project Name> PLC > PLC Project Name> References, and right click and click “Add Library”. Search for the “Tc2\_ModbusSrv” and add it.





Next import the downloaded POUs and DUTs into your program.

The blocks work as follows:

FB\_ModbusPoll - will poll the Modbus server at a certain start address and output the value.

FB\_CSV\_Writer - will write 2 pieces of data to a CSV file using a rising edge input.

FB\_ReadModbus - can poll 2 devices simultaneously when called in main it also will export the data automatically to a CSV file.

HexStringToReal – is a helper block to turn the raw Modbus data into a 32 bit floating point number.